Engineering Automated Operations for NASA's Deep Space Network

Eva Bokor, Patricia Santos, Paul Pechkam, Marla Thornton, Patrick Olguin,
Bryan Camilli, Manuel Gomez

Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109

Abstract

The NASA Deep Space Network is unique among space communication systems. The antennas and signal processing devices must be individually configured and calibrated before each spacecraft tracking or commanding activity. This preparation requires hundreds of keystrokes as operators issue directives to the receivers, transmitters, and other devices. Like all computer keyboard activities, this is error prone compared to an automated function, described here, that reduces both labor required and clerical errors. We place a special focus on difficult areas that required intensive engineering effort.

Keywords

automation, temporal dependency network, spacecraft operations

Introduction

The NASA Deep Space Network, or DSN, is an international network of spacecraft tracking stations. The stations are located at Goldstone, California, Madrid, Spain, and Canberra, Australia. Regardless of the location of a spacecraft in the sky, one or more of the DSN's antennas can observe it. Each of the antennas is precisely steered to remain pointed at the spacecraft as the earth rotates. The network supports two-way communication with interplanetary space missions and some Earth-orbiting missions, including emergency support of the Space Shuttle. The Jet Propulsion Laboratory manages and operates the network for NASA from a central operations center in Pasadena, California.

The DSN is unique among space communication systems in that it is link configurable. This means that for each pass of a spacecraft, a unique collection

of antennas, receivers, transmitters, and other signal processing devices can be assigned to support tracking. It is this capability to "mix and match" that creates the need for a high level of human effort by the station operators. The level of effort required, of course, varies widely with the type of activity. Staffing, however, is driven by the activity peaks that occur at launches, planetary encounters, and landings. To reduce these costs and improve efficiency, the Jet Propulsion Laboratory began exploring means for automating routine and repetitive activities in the late 1980's.

Initial efforts were based on software macros. These collected sets of directives or commands and executed them as a script. This approach has three significant limitations: 1) the inability to explicitly specify all possible outcomes, 2) a lack of visibility into system status if the macro fails while processing[1], and 3) a lack of flexibility in extracting data and using it to populate variables. To overcome these limitations, we began to explore a more sophisticated approach.

The next step in implementation was to develop a representation of the human operator's knowledge and actions that could be used by a computer, and then to acquire that knowledge. A successful prototype led to approval for full-scale implementation. Performance, however, was marginal, for reasons that will be discussed later. A second effort resulted in a fully functional system. The following sections will describe the development of the knowledge representation, knowledge engineering, software implementation, and operational use of this approach to DSN automation.

Knowledge Representation

To fractionate monitor and control activities, four scenarios for routine pre- and post-track operations were developed. These are:

Tracking - Measures the angular position and velocity of the spacecraft.

Tracking and Telemetry - Adds one-way down-link communication from the spacecraft.

Tracking, Telemetry, and Command - Adds up-link communication with the spacecraft.

---

[1] Cooper, Lynn P., "Operations Automation Using Temporal Dependency Networks," Proceedings of the Technology 2001 Conference, San Jose, CA, December 3-5, 1991.

Tracking, Telemetry, Command, and Ranging - Adds measurement of the distance of the spacecraft from earth.

For any of these four scenarios, the actual equipment used, both hardware and software, may vary greatly depending upon the spacecraft and antenna involved. Our implementation takes considerable pains to hide this fact from the operator. The automation assembly presents generic activities using generic equipment. These generic activities are represented, both graphically and conceptually by temporal dependency networks (TDNs).

A TDN is a directed acyclic graph containing the temporal and behavioral knowledge required to perform a specific task. A sample TDN is illustrated in Figure 1. Each arc within the graph represents a precedence relationship. Each node represents a discrete task or event that must be performed or deliberately omitted (granted an exception by the human operator) before the graph can be further traversed. To further explore TDNs as a knowledge representation technology we must examine the contents of the nodes, which are called *blocks* in this application.

Blocks consist of a related set of directives and conditions, as the fundamental element of control in the DSN. A directive is analogous to a command in UNIX or DOS. It is a sequence of characters terminated by a carriage return. Human operators, working at computer consoles, issue directives and receive confirmation of their execution. The confirmation may be a direct response by the receiving system, a text message, or a change in a data field displayed on the console. This combination of directive-response is known as positive closed loop control. Some of the equipment does not have this feature, a fact that adds to the challenge of automation. In general, when positive closed loop control is lacking, the automation must be interrupted and a manual operation is performed to confirm execution of the directive.

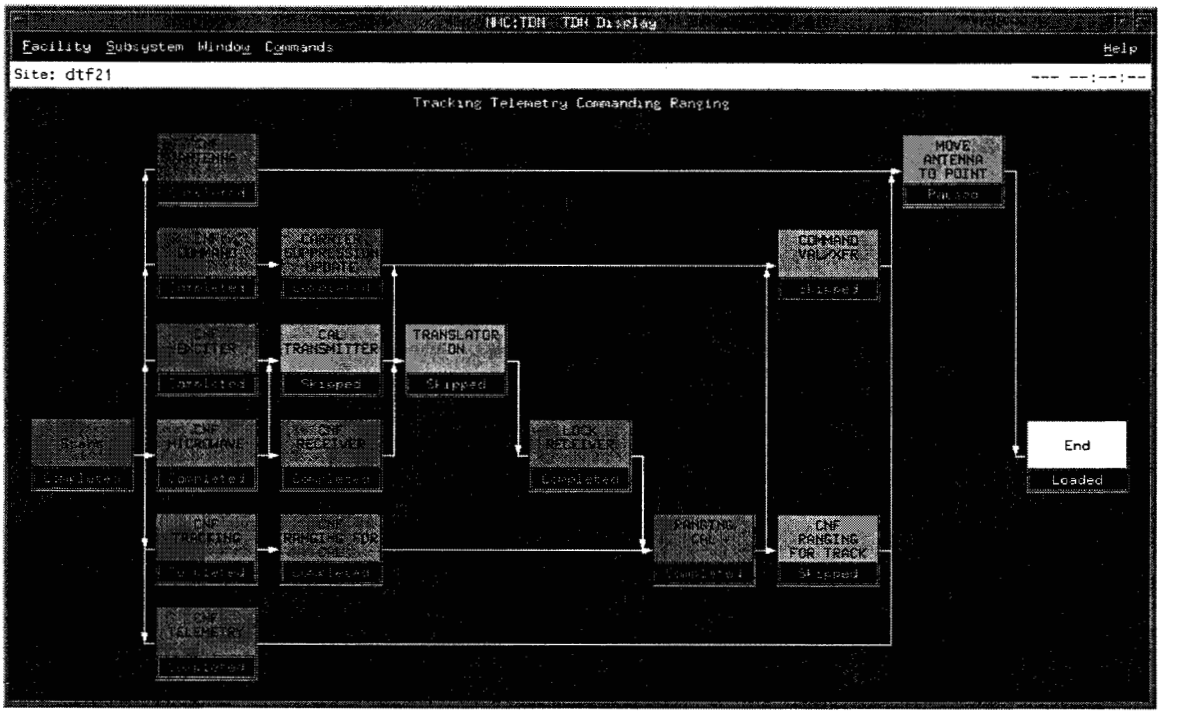

Figure 1. Temporal Dependency Network

A block consists of sets of directives that, together, perform a discrete task. This task may require one or more devices to be controlled and to interact. The names of the blocks indicate their purpose. Typical block names include:[2]

Configure Receiver
Move Antenna to Point
Stow Antenna

Along with directives, the blocks contain pre-conditions, which are entry criteria that must be satisfied before the block can be executed, and post-conditions, which are exit criteria for the blocks.

To describe the monitor and control of the DSN at an abstract level, the activities are divided into pre-track, track, and post-track phases. Pre-track, typically the most labor intensive, includes allocation, configuration, and calibration of equipment and movement of the antenna to point at the spacecraft's position. When the radio link with the spacecraft is acquired, track phase begins. While pre-track activities may take 30-60 minutes, track phase may last for hours, and even tens of hours. If all goes well, this phase consists

[2] Note that while the block names imply interaction with a single device, in fact, several devices may interact with a given block.

of merely monitoring data flows. Nevertheless, if problems occur or sophisticated activities are scheduled, this phase may contain the most complex human-computer interactions. The post-track phase begins when the spacecraft goes down over the horizon and communication is lost. The operator then returns the equipment to the available pool and returns the antenna to its stowed position.

Knowledge Engineering

Knowledge engineering is the process of capturing human expertise and transferring it into a form that is usable by a computer program. On this project, this consisted of interviews with operations personnel and with the engineers that built the various devices of the DSN. The knowledge they supplied helped develop the TDNs and the information contained in the individual blocks. Finally, programmers convert the information into code.

Several challenges were encountered during the knowledge engineering. The first was identification of the appropriate knowledge sources. No unified data source listed the names of the engineers who built the devices. In some cases, the person had left the Laboratory and maintenance had been handed off to a person who lacked in-depth understanding.

After the experts were identified, extracting the knowledge provided additional challenges. Often the engineers responsible for the devices, and the operators who used them daily, disagreed on the proper operational procedures. Differences also existed between individual operator's views of the devices. Attempts to resolve the differences with serial meetings were unsuccessful. We found it more effective to bring all the parties into a meeting and distill a consensus. It was extremely helpful to have a senior moderator present at these discussions.

In some instances, issues could only be resolved by "touching the elephant." Because the nearest tracking station is a three hour drive from the Laboratory, much of it on two-lane desert roads, we encountered significant logistic difficulties.

Three Key Challenges

Three characteristics of the DSN make automation particularly challenging. The first is the need to configure the equipment string for each spacecraft pass, as described above. Another is the lack of pre-automation preparation.

In the classical approach to process automation, a prerequisite step is pre-automation, in which the process is completely debugged and the workspace made as easy as possible for the human operator.[3] Only then is automation applied to the problem. Financial and technical considerations prevented this prerequisite step from occurring. Much of the DSN's equipment is a diverse collection of aging, custom-built devices.[4] Elimination of equipment would mean loss of valuable science data as they are still used to communicate with older spacecraft. Replacement would be prohibitively expensive. The result is that we worked with systems that were not, strictly speaking, ready for automated control.

A third complication came from the requirement that the tracking station operator must be able to modify the automated monitor and control software at any time during around-the-clock operation. This requirement is driven by the exigencies of the DSN. The unique demands of monitor and control of planetary spacecraft imply frequent last-minute changes and work-arounds. Thus the logic of the automation may need modification when software professionals are not available. A significant part of the logic must be coded in a scripting language to permit local, real-time modifications. Other key information is kept in tables that can be modified with a text editor.

Preliminary Implementation

After the successful prototype, an operational implementation was created as a part of an upgrade to the monitor and control workstation. Unfortunately, performance of the TDN automation portion of the system was marginal and it was never used for tracking spacecraft. As discussed above, the blocks were written in a scripting language to facilitate rapid change by persons who are not professional programmers.

---

[3] Schonberger, Richard J., and Edward M. Knod, *Operations Management*, 5th edition, Irwin, Burr Ridge, IL, pp. 79-80.

[4] Some of the computers are over 15 years old, while some mechanical devices are 30 years old.

The first implementation used a product called BasicScript[5] as the scripting engine. This was a poor choice for this application, but that was not evident until implementation was nearly complete. Each of the approximately 36 scripts resided in a separate file on disk. The language requires that the entire application program must reside in a single file. This meant that repetitive functionality encapsulated in a subroutine needed an instance for each script. This created severe configuration management problems. In addition, the programmers pushed the edge of the performance envelope. Some features of the language, such as limits on the number of variables and the lack of a function primitive made the implementation difficult.

Nevertheless, in building this version, the developers learned a great deal about the actual requirements and potential design difficulties in the application domain. The first version of the TDN Engine became the classic "throw one away" described by Brooks.[6]

Final Implementation

In the second and final implementation, tool command language,[7] or Tcl, is the scripting language. The source code for the Tcl interpreter is freely available, so we were able to extend the language to meet our needs. Several constructs were added. The resulting language is known as the automation language for managing operations or ALMO. Although ALMO is a superset of Tcl, the extensions permit the scripts to be written in a syntax that is very close to the macro language used in the previous monitor and control system. This makes it easier for tracking station operators to assist in script writing and maintenance.

Software Architecture

The software collectively known as the Automation Assembly consists of three processes. These are the TDN Engine, the Block Engine, and the Block Manager. The software architecture is illustrated in Figures 2 and 3. The TDN Engine controls execution of the TDN and is controlled by the monitor and control workstation. The Block Engine, which is controlled by the TDN Engine, interprets the ALMO, and interacts with the various

---

[5] Basic Script is a product of Summit Software, Jamesville, NY.
[6] Brooks, Frederick P., *The Mythical Man-Month*, Addison-Wesley, Reading, MA, 1975, p. 116.
[7] Osterhout, John K., *Tcl and the Tk Toolkit*, Addison-Wesley, Reading, MA, 1994.

telecommunications systems. The Block Manager aids in control of the multiple versions of TDN blocks and in selection of the appropriate block for each tracking instance. The functions of the blocks are further explained in the next section which contains a narrative of the automated tracking activity.

Spacecraft Tracking Events

Shortly before a monitor and control workstation is scheduled to perform a track, all necessary data is downloaded to the workstation. This includes the identity of the spacecraft to be tracked, the designated antenna, and information known as the sequence of events (SOE). The SOE contains the what, when, and how of each tracking activity. It specifies the equipment to be used, the time and point of spacecraft rise and set, and parameters needed for equipment calibration. In addition, it specifies the version of the device controller software to be used.

All of the signal processing devices in the DSN are computer controlled. This permits use of different software versions to achieve different technical objectives within the same hardware. Changing the version of software invoked provides the option, for example, of using any one of several versions of a transmitter for a given tracking activity. The flexibility inherent in this technique creates an additional challenge for the Automation Assembly.

The TDNs are written at an abstract level for generic devices and do not have awareness of the multiple versions that exist in practice. Therefore, the Automation Assembly requires the intelligence to select the proper version of the signal processing device. The Assembly uses an internal table to select the correct versions of the device controller's software.

After determining the appropriate block and versions, the TDN Engine uses a library function to call the Block Manager. The Block Manager responds with the files containing the blocks. The TDN Engine creates an instance of the Block Engine for each block. These processes load the block file into memory and notify the TDN engine when complete. The selected TDN is now ready for execution. A window showing all key parameters is displayed to the human operator for approval. When approved, the TDN may be executed.

The operator clicks the *start* block and each of the initial blocks (those with *start* as their predecessor) begin execution. If the block completes successfully,

the successor block is started. Note that multiple blocks may have a single predecessor. This provides opportunities for parallelism. If a problem occurs during block execution, the operator is prompted to intervene. He or she may correct the problem and restart the block, choose to continue in spite of the problem, or abort the block. When the terminal block *end* is reached, the activity is completed. The operator can now resume manual control or start another TDN.[8]

Conclusion

Although the DSN provided unique challenges, preliminary reports indicate that the effort is successful. The Automation Assembly is now in the final stages of testing. Results indicate that time to conduct pre-track activities is reduced by 50% or more. In one test, an operator was able to manage two spacecraft at the same time, something that is not possible without automation. Nevertheless, automation is not a substitute for an experienced DSN operator. When serious anomalies occur, the expertise of the human is needed to correct the problem or devise a work around.

Acknowledgments

The authors would like to thank Sharon Anthony, Tim Gregor, and John Jansson of the Goldstone Deep Space Communication Complex (DSCC), Ossi Larikka and Michael Smith of the Canberra DSCC, and Francisco Jimenez of the Madrid DSCC for their significant contributions to this effort. We also thank Hugh Henry, Supervisor, Network Monitor and Control Group, Jet Propulsion Laboratory, for his assistance in the writing of this paper.

The research described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology, and was sponsored by the National Aeronautics and Space Administration.

[8] Note that blocks can also be executed in a standalone mode, independent of TDNs.

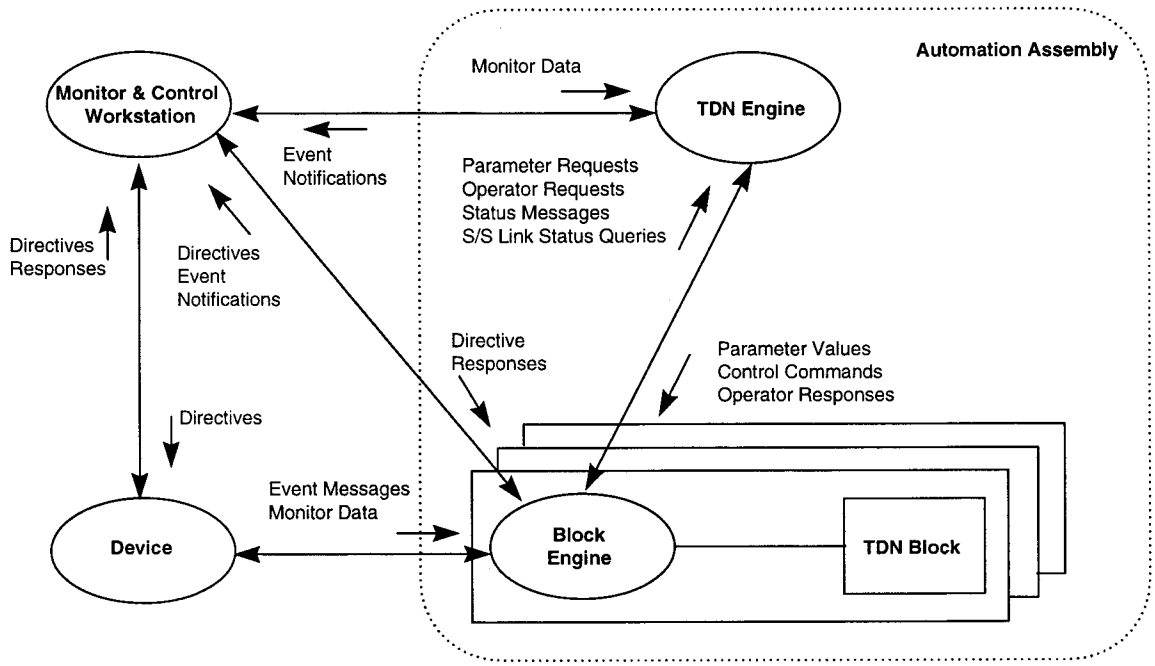

Figure 2. Runtime Software Architecture

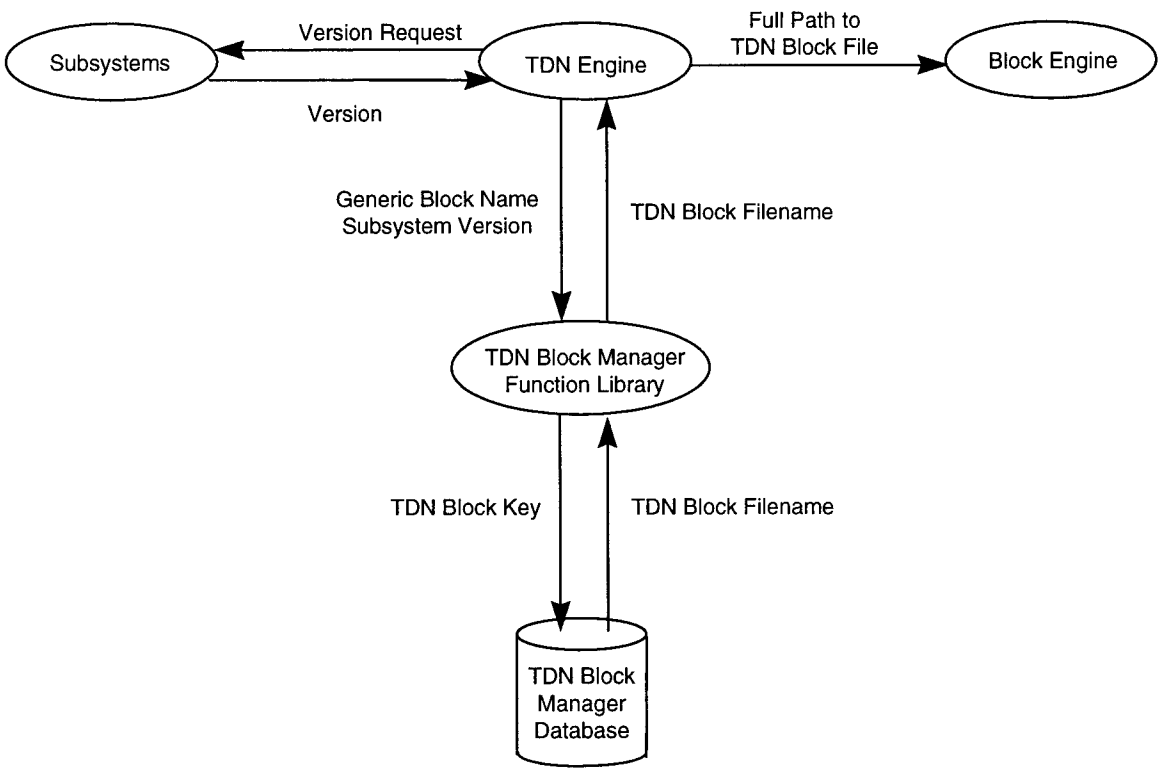

Figure 3. Block Management Architecture